Action Items

•••

Benedikt Hegner
(CERN EP-SFT)

Gaudi Workshop
23.9.2016

My conclusions so far

We started a discussion to see what we need to re-design of Gaudi to prepare it for the future.

In the end, we don't turn Gaudi upside down, but some aspects become more important and clearer now

$\Rightarrow$ new patterns emerge (e.g. Gaudi::functional)

Overall Gaudi design is still state-of-the-art...

My conclusions

We started a discussion to see what we need to re-design of Gaudi to prepare it for the future.

In the end, we don’t turn Gaudi upside down, but some aspects become more important and clearer now

$\Rightarrow$ new patterns emerge (e.g. Gaudi::functional)

Overall Gaudi design is still state-of-the-art…

… while parts of the implementation clearly aren’t

So what did we achieve

- We made a good progress in understanding the different use cases
  - Sometimes not clear what is legacy and what are real boundary conditions
  - I would however consider the use-case document as almost ready

- We were talking a lot about chances of unification and simplification
  - In sometimes lengthy, but always constructive discussions!

- Had to cover too many topics, thus lots of only half-conclusions
  - Requires follow ups in Gaudi Developers Meeting

Compiled a few potential action items, which we should trace in JIRA

Data Handles

- We seem to have consensus on:
  - Dropping update semantics
  - Different types for `ReadHandle<T>`, `WriteHandle<T>`
  - Reentrant implementation
  - Explicit `get()` and `put()` calls

Action item 1:

Write down and implement new data handle interface - both C++ and config side

Best if we continue today!

Event Store

- Having users going via handles limits required interface
- A good time to re-evaluate what has been provided so far
- Quite a chance for unification

Action item 2:
Define new data store interface incorporating experience from StoreGate and DataSvc

Best if we continue today!

Modernizing code

For the next year we’ll try to be more aggressive on starting and completing migrations. A few candidates:

Action item 3:
Finish interface of new property declaration

Action item 4:
Drop interface versioning

Action item 5:
Drop custom reference counting

Identifying dead code

For some components we weren’t sure who actually uses them and whether we still need them

Action item 6:
Assess which components are used by which collaboration

Action item 7:
Drop zombie components

Simplifying development and compilation

Developers would like to be able to compile Gaudi on their own machine

Action item 8:
Port Gaudi to Mac

Action item 9:
Port Gaudi to Ubuntu

Action item 10:
Separate optional subsystems and introduce configuration switches

Pushing components upstream

Various problems have been solved independently in different collaborations. We've learned about a few this week.

Action item 11:
Follow up on trigger config persistency solutions

Action item 12:
Check whether ATLAS' THistSvc can replace the AIDA-based one

Gaudi::functional

- Broad consensus that the functional approach is a good paradigm
- Non-intrusive / “simple” add on
- Scary template syntax - deserves particularly detailed documentation

Action item 13:
Integrate Gaudi::functional with examples and docs.

Documentation

Everybody agrees that we need better documentation and examples.

Action item 14:
Lay out structure for Gaudi introductory course and documentation

Need more discipline:
No merge request review without looking at documentation

Conditions System

One of the more controversial discussions:

- How to avoid memory explosion / do garbage collection ?
- Retrieve upfront or on demand?
- How and what to schedule

Action item 15:
Investigate in code whether a common ICondSvc and Handle interface is possible

Python Config Interface

- Experiments suffer from complex configs
- Could the Gaudi config system help in proposing proper “pattern”?
- Dropping sequencers control flow needs to be configured explicitly

Action item 16:
Check in ATLAS whether the Python control flow syntax covers the use cases

Action item 17:
Extract examples of higher-level config concepts Gaudi could provide

Distributing the work

This already is a long list of action items, but...

- How do we want to tackle them?
- By when do we want to have them addressed?
- What's their priority?
- Who can actually work on it?

In the following slide I at least noted down who candidates for the various tasks **could be**

This is a proposal, not an assignment of work!

Strawman Plan 1/2

Action item 1 - new DataHandle interface (Charles, Marco, Benedikt)
Action item 2 - new Data store interface (StoreGate + DataSvc expert)

Action item 3 - finish interface of new property declaration (Marco)
Action item 4 - Drop interface versioning (anyone)
Action item 5 - Drop custom reference counting (anyone)
Action item 6 - Assess which components are used by which collaboration (experiment expert)
Action item 7 - Drop zombie components (anyone)
Action item 8 - Port Gaudi to Mac (Attila, Joschka)
Action item 9 - Port Gaudi to Ubuntu (Benedikt, Marco)
Action item 10 - Separate optional subsystems (CMake ‘expert’)

Strawman Plan 2/2

Action item 11 - Follow up on trigger config persistency solutions (Frank, Gerhard)
Action item 12 - Check ATLAS' THistSvc (LHCb expert)
Action item 13 - Integrate Gaudi::functional with examples and docs. (Gerhard)
Action item 14 - Lay out structure for Gaudi introductory course and documentation (Simon)
Action item 15 - Investigate common ICondSvc and Handle interface (Hadrien, Benedikt)
Action item 16 - Check control flow syntax with ATLAS (Frank, Benedikt)
Action item 17 - Higher-level config concepts Gaudi could provide (experiment experts)